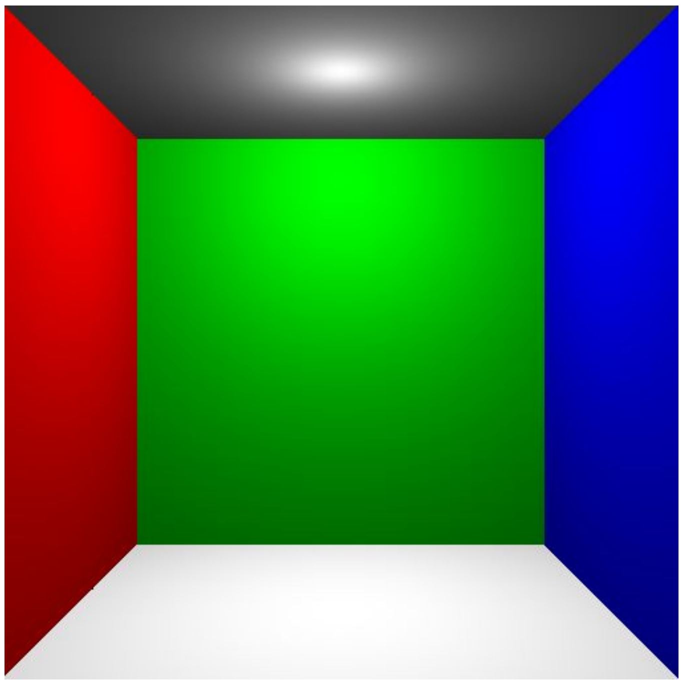
Computer Graphics - Final Project Report
Photon Mapping

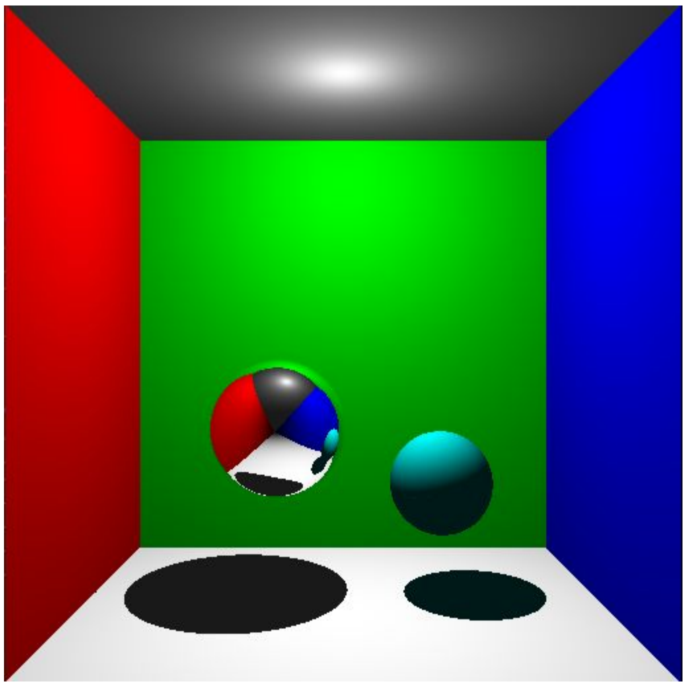
Mayank | Piyush | Suryansh

Motivation and Background:
We started our project with the idea of implementing special effects like refraction, reflection, shadow, bleeding etc with multiple objects inside a cornell box. Since we had already worked on Ray tracing in our Project 2 as part of the curriculum of CAP 5705, we started with implementing Ray tracing.

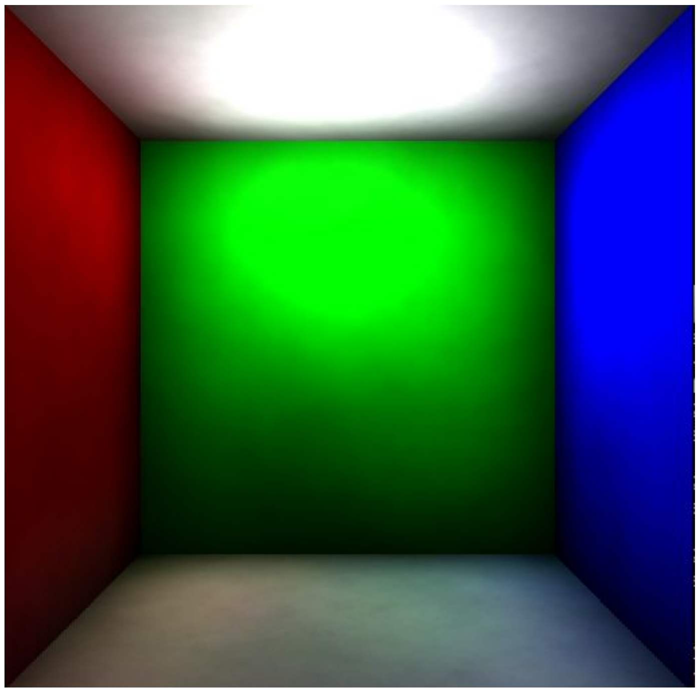
Ray Tracing is a technique for generating an image by tracing the path of light through pixels in an image plane and simulating the effects of its encounters with virtual objects. It is capable of producing a very high degree of visual realism, but at a greater computational cost. Ray Tracing is best suited for applications where the image can be rendered slowly ahead of time, such as in still images in our project, and poorly suited for real-time applications like video games where speed is critical.

After implementing Ray tracing, we were able to generate effects like reflection, refraction and shadows inside the cornell box but we were not satisfied with the realism of rendered images. This was because of the **limitation** of Ray tracing where it was not practical to simulate diffuse bounces when the ray leaves the surface in a random direction, simply because the light's rays are traced backwards from the eye to the light. This makes ray tracing limited in the images it can produce. Any effect that involves a diffuse bounce, like color bleeding, before the light reaches the eye is not possible in standard ray tracing. Also, the ray will come out of the same point it reaches the surface i.e. no successions of micro bounces will result into it leaving the surface at any other point. Below are some images representing our implementation of ray tracing inside a cornell box:


Ray Tracing Cornell Box


Refraction, Reflection and Shadow Effect

Then we incorporated Photon mapping in our project. Photon mapping is an elegant way to solve the above limitations of Ray tracing. Before any standard Ray Tracing is done, light (photons) is sent from the light sources in the scene and allowed to bounce around, then stored in a "photon map". The photon map stores final positions of the photons as well as color and incoming direction. After the map is built, standard Ray Tracing is done, and for each bounce the photons near the location of the bounce are essentially treated as small light sources. This allows us to generate complete light paths that would not be normally considered.
It also helped us simulate diffuse interreflections by allowing the light to bounce around the scene, grabbing color at each step. To perform this, photons were stored at each diffuse bounce, but were still allowed to continue on their path, until they got absorbed. Below is an image representing photon mapping inside a cornell box:



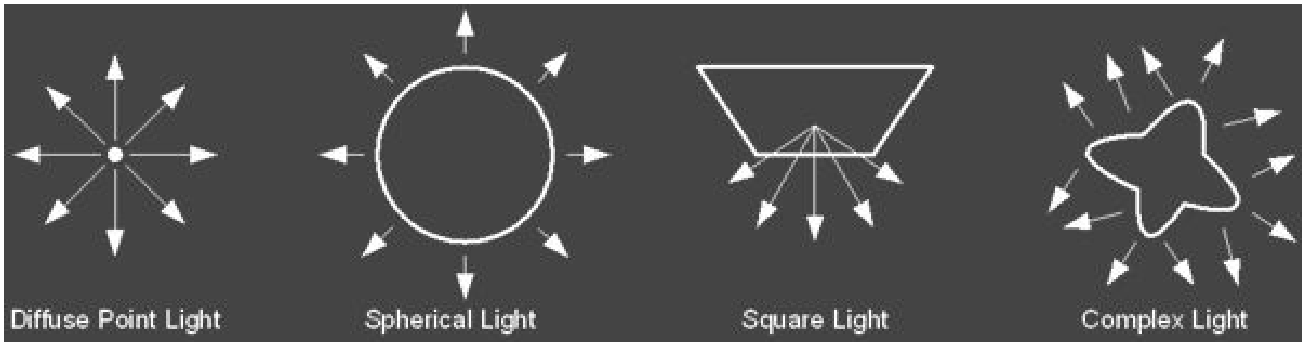
Photon Mapping Cornell box with color bleeding

Implementation:
Our idea for implementing the above mentioned photon mapping in our project was to decouple the representation of a scene from its geometry and store the illumination information in a global data structure which is the photon map. We utilised the two pass algorithm, where the first pass builds the photon map by tracing photons from each light source. And the second pass renders the scene using the information stored in the photon map.

First Pass of the algorithm

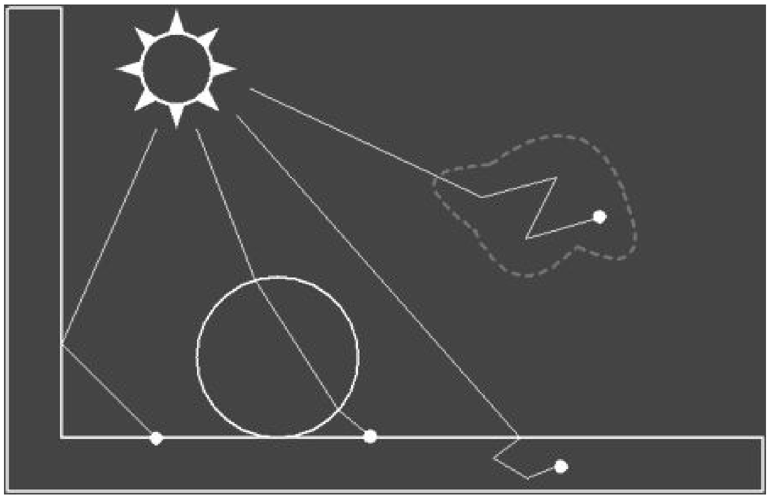
The first pass of the algorithm consist of 3 steps:

1. Photon emission: It is the process of emitting discrete photons from the light sources and tracing them through the scene. The primary goal of this pass is to populate the photon maps that are used in the rendering pass to calculate the reflectance at surfaces and out-scattered radiance in participating media. When a photon hits an object, it can be reflected, transmitted, or absorbed.



Different types of light sources

2. Photon scattering : The emitted photons from light sources are scattered through a scene and are eventually absorbed or lost. When a photon hits a surface we can decide how much of its energy is absorbed, reflected and refracted based on the surface's material properties.

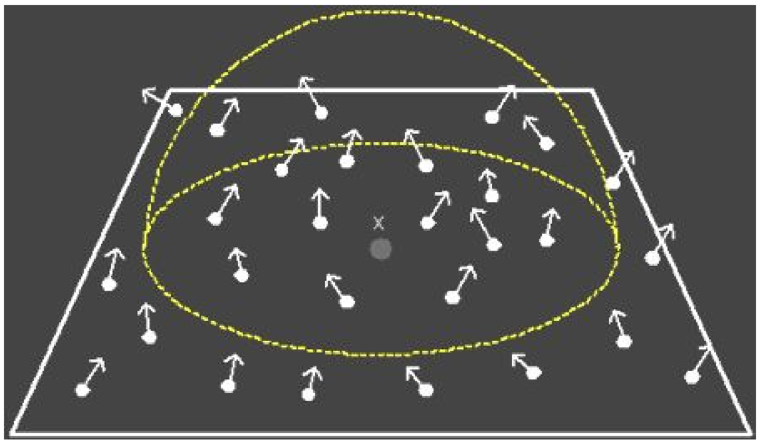


Photon Scattering

3. Photon storing: We stored the position, color and direction of photons in the scene. For this we required a good data structure as it was necessary and important to locate photons in the neighborhood of the ray-intersection.

Second Pass of the algorithm

In the second pass of the algorithm we rendered the scene with the help of the photon map built in the first pass. In this step of the algorithm, the photon map created in the first pass is used to estimate the radiance of every pixel of the output image. For each pixel, the scene is ray traced until the closest surface of intersection is found. At this point, the rendering equation is used to calculate the surface radiance leaving the point of intersection in the direction of the ray that struck it.



Photon Rendering

Ray-Sphere and Ray-plane Intersection

We have spheres and planes as different objects in our project, so we implemented Ray sphere and Ray plane intersection to find the points of intersection. To find the Ray sphere intersection, we have the equation of sphere as $x^2+y^2+z^2=R^2$, where x,y,z are the center and R is the radius. And the equation of ray as p + tD where p is origin, t is the distance and D is direction. We solved the above 2 equations to get the point of intersections. For Ray-plane intersection, we have the equation of ray as p + tD where p is origin, t is the distance and D is direction and equation of plane as P.N + d. We solved both equations to get the point of intersection.

Special Effects

1. **Reflection:** For implementing reflection, we used mirror reflection. We knew the incident ray, the normal at the point of intersection, and thus computed the reflected ray.
2. **Refraction:** For implementing refraction, we used Snell's law, which is used to describe the relationship between the angles of incidence and refraction, when referring to light or other waves passing through a boundary between two different isotropic media.
3. **Soft Shadow:**The common-sense notion of shadow is a binary status, *i.e.* a point is either in shadow or not. This corresponds to hard shadows. However, point light sources do not exist in practice and hard shadows give a rather unrealistic feeling to images. In the more realistic case of a light source with finite extent, a point on the receiver can have a partial view of the light, *i.e.* only a fraction of the light source is visible from that point. Also ambient light or other light sources also contribute to this, thus creating soft shadows. This effect is performed by scattering Photons all over the Cornell Box and then calculating the shadow based on the Photons.
4. **Color Bleeding:** Color bleeding is the phenomenon in which objects or surfaces are colored by reflection of colored light from nearby surfaces. This effect is performed by the colored reflection of indirect light.

5. **Object movement:** Each time an object is moved inside the cornell box, the whole scene is re-rendered, to implement real time object movement.
6. **Light source movement:** The same idea was also implemented for the light source.

Challenges:

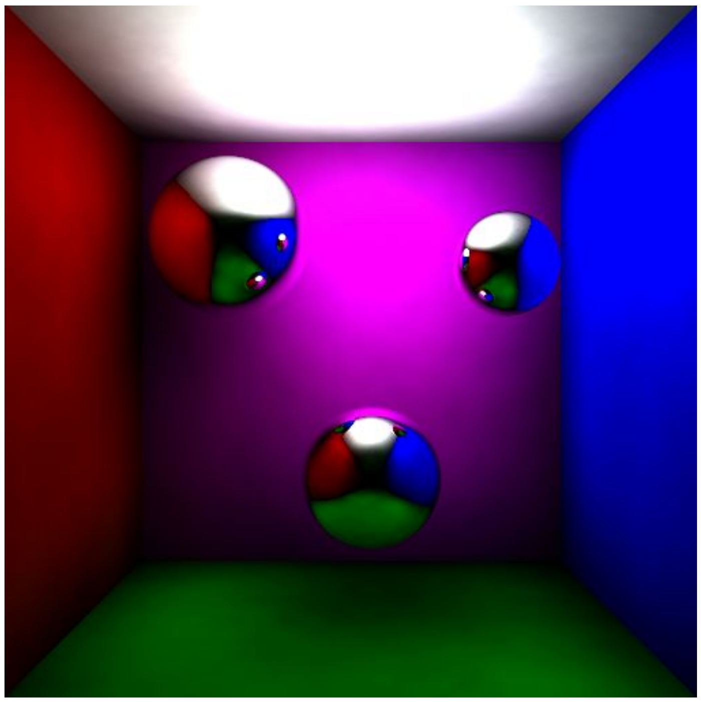
We faced a lot of challenges in this project because of its high complexity.

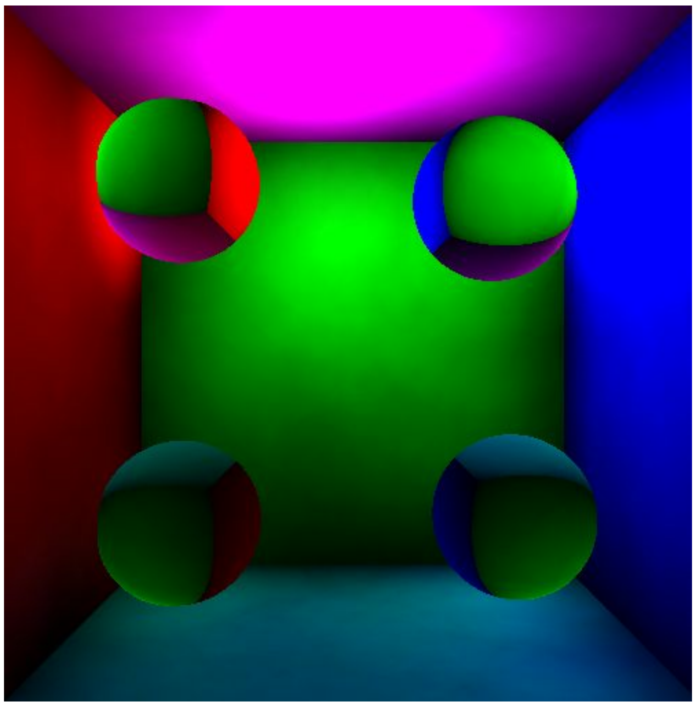
1. Since photon mapping is very heavy on the computation side, we had to keep our pixel density of the rendered images to a moderate amount(512 X 512 pixels) to be able to generate quick results.
2. One major challenge that we faced was the termination of photons i.e to bring their energy to zero or almost zero. We kept on reducing the energy of the photon after every bounce but were not able to completely negate it. In the end we came with a termination condition dependent on the number of bounces rather than the energy of each photon.
3. We faced a considerable amount of challenge in storing the photons to create a photon map (in the first pass of our algorithm), and had to design our own Data Structure to store the photons.
4. Implementing Refraction was a challenge for us.
5. Implementing Color Bleeding and Soft Shadow effect through photon mapping had a steep learning curve, and took us quite a lot of time.

Results and Conclusions:

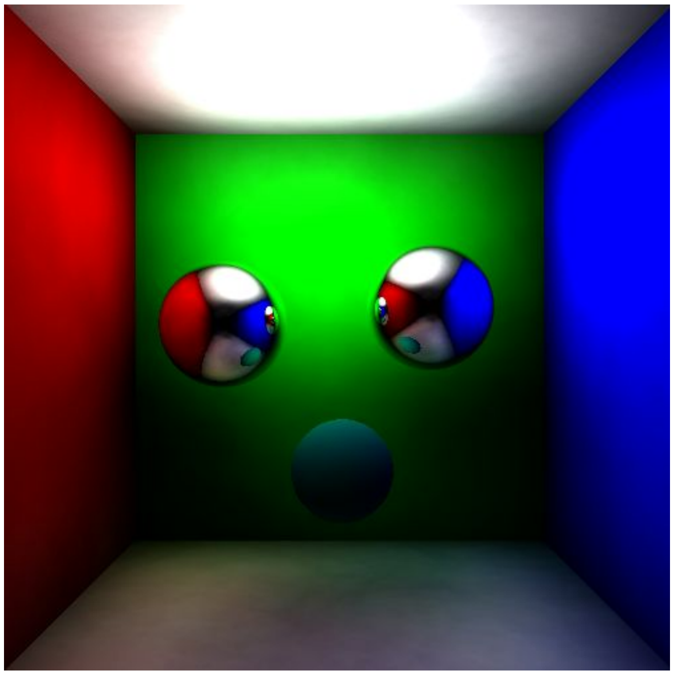
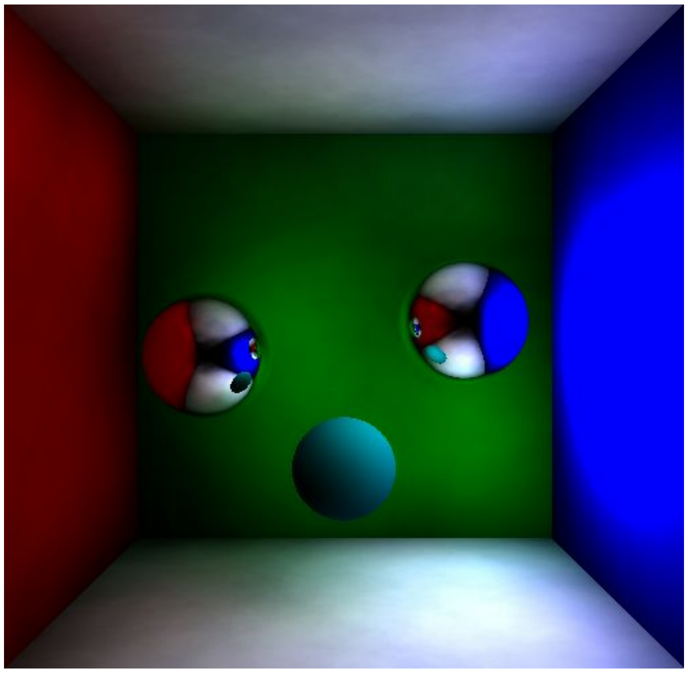
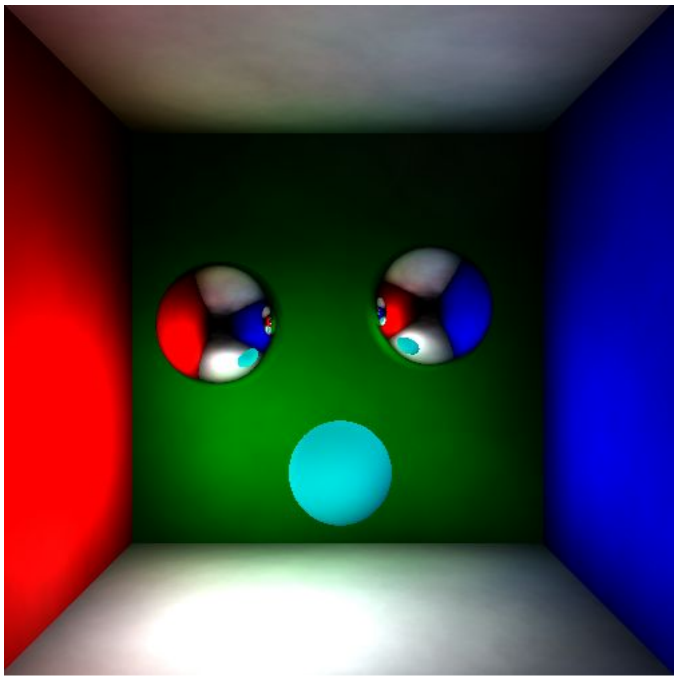
With the implementation of photon mapping, we got interestingly realistic images of the objects inside the cornell box.

1. Below two images shows the effects of reflection, refraction, shadow and color bleeding in our result.

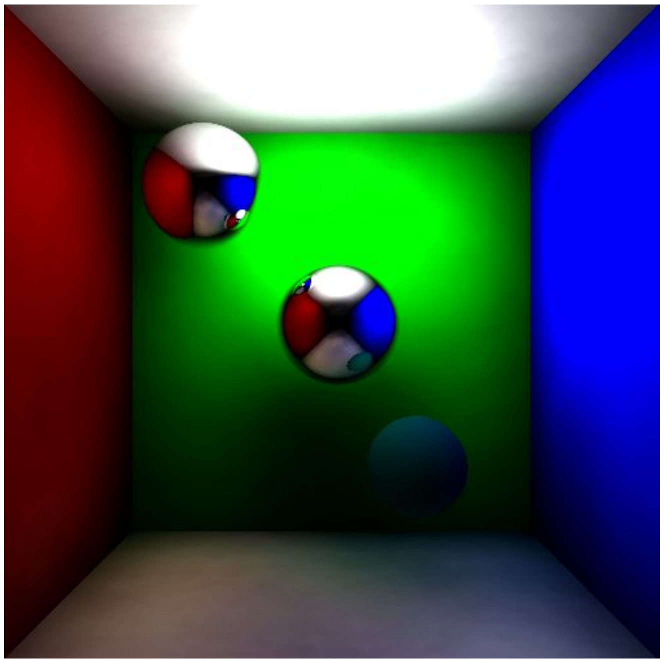
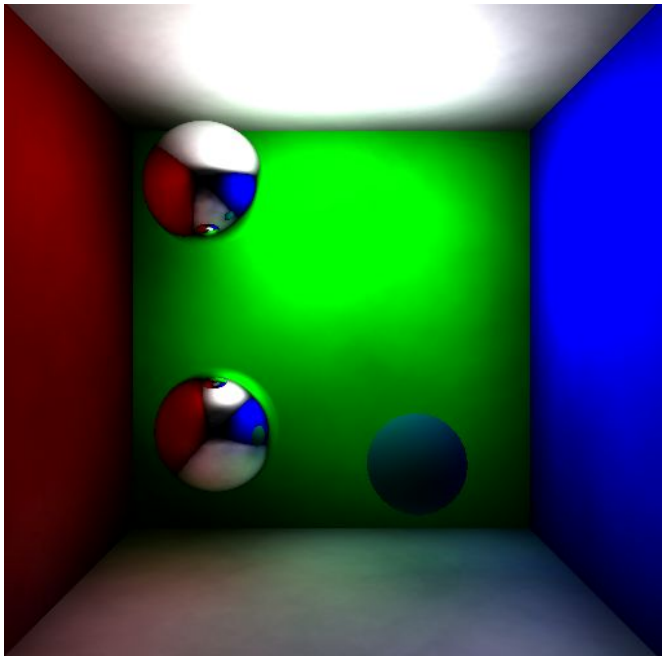
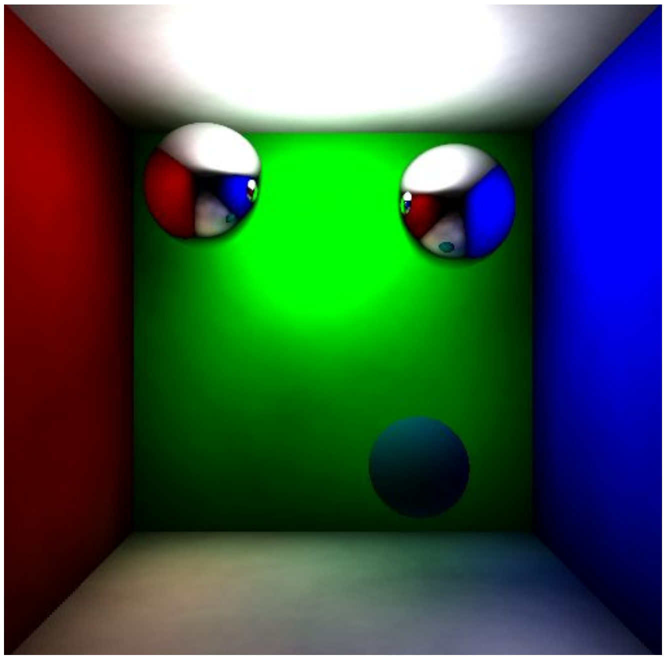




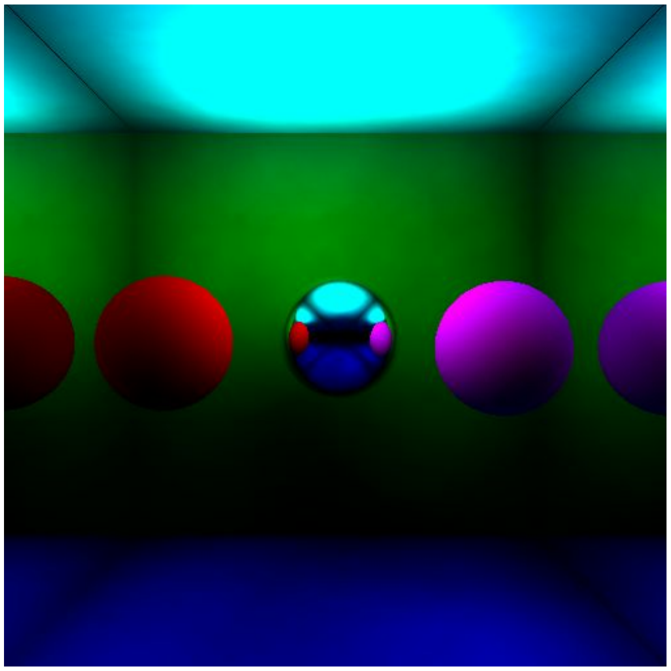
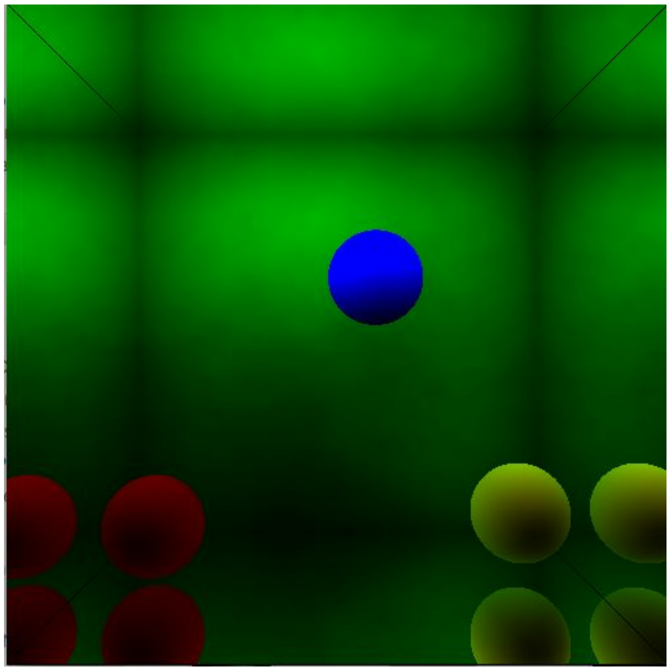
2. Below 3 images show the movement of light source from top wall (in first image), to right wall (in second image), to bottom wall (in the third image). This is real time movement of light source, and how we rendered the resulting image.

3. Below 3 images show the movement of objects inside the cornell box and how we rendered the resulting image.

4. Below 2 images show our additional work, where we created a mirror effect inside the cornell box.

References

1. http://graphics.ucsd.edu/~henrik/papers/photon_map/global_illumination_using_photon_maps_egwr96.pdf
2. http://marctenbosch.com/photon/
3. https://graphics.stanford.edu/courses/cs348b-01/course8.pdf
4. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.300.6472&rep=rep1&type=pdf
5. http://graphics.ucsd.edu/papers/photongfx/photongfx.pdf

Acknowledgement:

Dr. Toler-Franklin for her guidance and assistance.